Perihelion advance

April 18, 2019

Abstract

Here we find the orbit of an object around a central mass in Newtonian and General Relativistic approach with and without the effect of discretization using Radau5 program in Fortran. First we derive the equations of motion in these two approaches then we discuss about the numerical results.

1 Newtonian gravity

Here first we derive the equations in Polar coordinate then we show the result of perihelion advance for two body problem in Newtonian gravity.

$$\frac{d^2\vec{r}_1}{dt^2} = \frac{-Gm_2}{|r|^3}(\vec{r}_1 - \vec{r}_2) \tag{1}$$

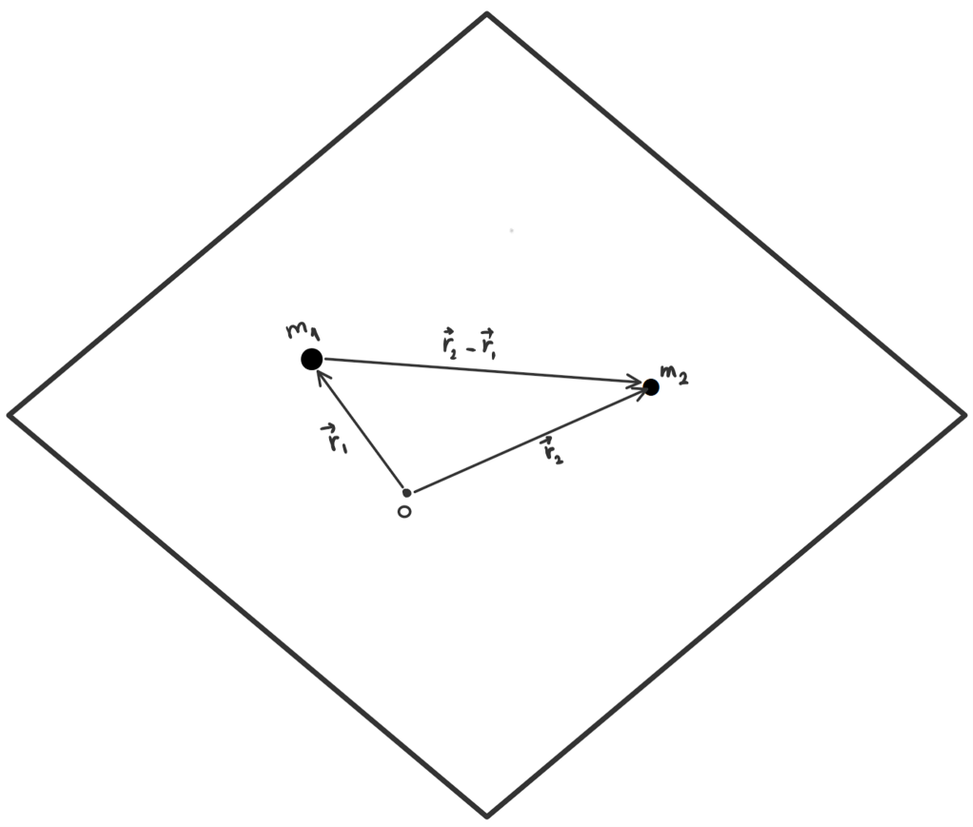
$$\frac{d^2\vec{r}_2}{dt^2} = \frac{-Gm_1}{|r|^3}(\vec{r}_2 - \vec{r}_1) \tag{2}$$

where $\vec{r} = \vec{r}_1 - \vec{r}_2$. Changing the coordinates to $\vec{r}$ and $\vec{R}_{com}$ we obtain the following equations,

$$\ddot{\vec{R}}_{com} = 0 \longrightarrow R_{com} = \dot{R}_{com}t + R_{com}(0) \tag{3}$$

The more interesting equation is,

$$\frac{d^2\vec{r}}{dt^2} = \frac{-G(m_2 + m_1)}{|r|^3}\vec{r} \tag{4}$$



Using the polar coordinate we have,

$$\vec{r} = r\hat{r} \longrightarrow \dot{\vec{r}} = \dot{r}\hat{r} + r\dot{\hat{r}} = \dot{r}\hat{r} + r^2\dot{\theta}\hat{\theta} \longrightarrow \ddot{r} = \hat{r}(\ddot{r} - r\dot{\theta}^2) + \hat{\theta}(2\dot{r}\dot{\theta} + r\ddot{\theta}) \tag{5}$$

So,

$$2\dot{r}\dot{\theta} + r\ddot{\theta} = 0 \longrightarrow \frac{d(r^2\dot{\theta})}{dt} = 0 \longrightarrow r^2\dot{\theta} = L = const \tag{6}$$

Moreover,

$$\vec{r} \times \dot{\vec{r}} = r\hat{r} \times (\dot{r}\hat{r} + r\dot{\theta}\hat{\theta}) = r^2\dot{\theta}\,\hat{r} \times \hat{\theta} = r^2\dot{\theta} = const \tag{7}$$

So we can solve the problem in 2 dimensions since the vector perpendicular to the surface including two masses is constant. The equation of motion for the relative vector reads,

$$\ddot{r} - r\dot{\theta}^2 = -\frac{GM}{r^2} \longrightarrow \ddot{r} - \frac{L^2}{r^3} = -\frac{GM}{r^2} \tag{8}$$

where $M = m_1 + m_2$. Changing time out for $\theta(t)$ and transforming $r(\theta(t))$ to $u(\theta(t)) = 1/r(\theta(t))$ we get,

$$u'' + u - \frac{GM}{L^2} = 0 \tag{9}$$

To obtain angle as a function of time we have another equation from conservation of angular momentum,

$$\frac{d\theta}{dt} = Lu^2 \longrightarrow t - t_0 = \int_{\theta_0}^{\theta} \frac{L}{u^2(\theta)}\, d\theta \tag{10}$$

For the purpose of our work we use the angle as a free parameter and also we do not solve the equation to obtain the time, since the angle is enough to obtain the perihelion and accepting it as a free variable helps us to simplify our problem.

1.1 Strong/weak regime way I

To go in different regime we define a parameter α which scales the physical quantities as following,

$$M \to \alpha M \tag{11}$$
$$v_{per} \to \sqrt{\alpha} v_{per} \tag{12}$$
$$T \to \frac{T}{\sqrt{\alpha}} \tag{13}$$

which are consistent with centripetal force formulas

$$v = \frac{GM}{R}, \qquad T = \frac{L}{v} \tag{14}$$

T is the object's period, v_{per} is the perihelion speed of the object and M is the central object's mass. By α we can go to the strong/weak limit while keeping one period orbit in time $\frac{T}{\sqrt{\alpha}}$.
The initial conditions for the problem are,

$$x_0 = R \tag{15}$$
$$y_0 = 0 \tag{16}$$
$$v_x = v_{per} \tag{17}$$
$$v_y = 0 \tag{18}$$

To compare the results with Mercury observations we use Mercury's facts

$$M_{mercury} = 3.285 \times 10^{24} \text{ kg} \tag{19}$$
$$T = 87.96 \text{ days} \tag{20}$$
$$v_{per} = 58.98 \ km/s \tag{21}$$
$$R = 46 \times 10^9 \ m \tag{22}$$
$$M_{sun} = 1.99 \times 10^{30} \ kg \tag{23}$$
$$\tag{24}$$

1.2 Strong/weak regime way II

The second way we can interpolate is by changing the distances and keeping masses constant as following,

$$M \to M \tag{25}$$
$$R \to \beta R \tag{26}$$
$$T \to \beta^{3/2} T \tag{27}$$
$$v_{per} \to \frac{v_{per}}{\sqrt{\beta}} \tag{28}$$

R is the perihelion radius. The initial conditions for the problem are,

$$x_0 = \beta R \tag{29}$$
$$y_0 = 0 \tag{30}$$
$$v_x = \frac{v_{per}}{\sqrt{\beta}} \tag{31}$$
$$v_y = 0 \tag{32}$$

We choose second way of rescaling since the distances are very important for us, because of another length scale in the problem which is lattice grids length.

1.3 code:

We use Radau5 code to solve the differential equation and getting the perihelion advance. In the code we solve the below equations,

$$u' = v_u, \quad v_u' = -u + GM/L^2 \tag{33}$$

the code for mercury like object is as following,

```
!#############################
  !FILE OPEN
  !#############################
  !!  open (unit = 12, file = "datajp.dat")

!#############################
  !End points
  !#############################
  ! --- endpoint of integration
  ! ------ obligatoire au debut -------
  xbeg=0d0
  xend=200d0
  !#####################################################
  !##########Initial values FOR SCHWARZCHILD,
  ! ##########we start from prehelion values
  !#####################################################
  !#### y(1) = u(ini) =1/R =0.021739130434782608695652173913043478260869565217391
  !#### u'(ini) = 0
  y(1)=1.d0/(46.d0)
  y(2)=0
  ! --- required tolerance
  !        rtol=1.0q-20
  !        atol=1.0q-7*rtol
  rtol=1.0q-22
  atol=1.0q-22
  itol=0
  ! --- initial step size
  h=1.0d-4

  real*16 rr,xx,yy
    real*16 twopi
    twopi=2.* 3.14159265358979323846264338327950288419716939937510q0
```

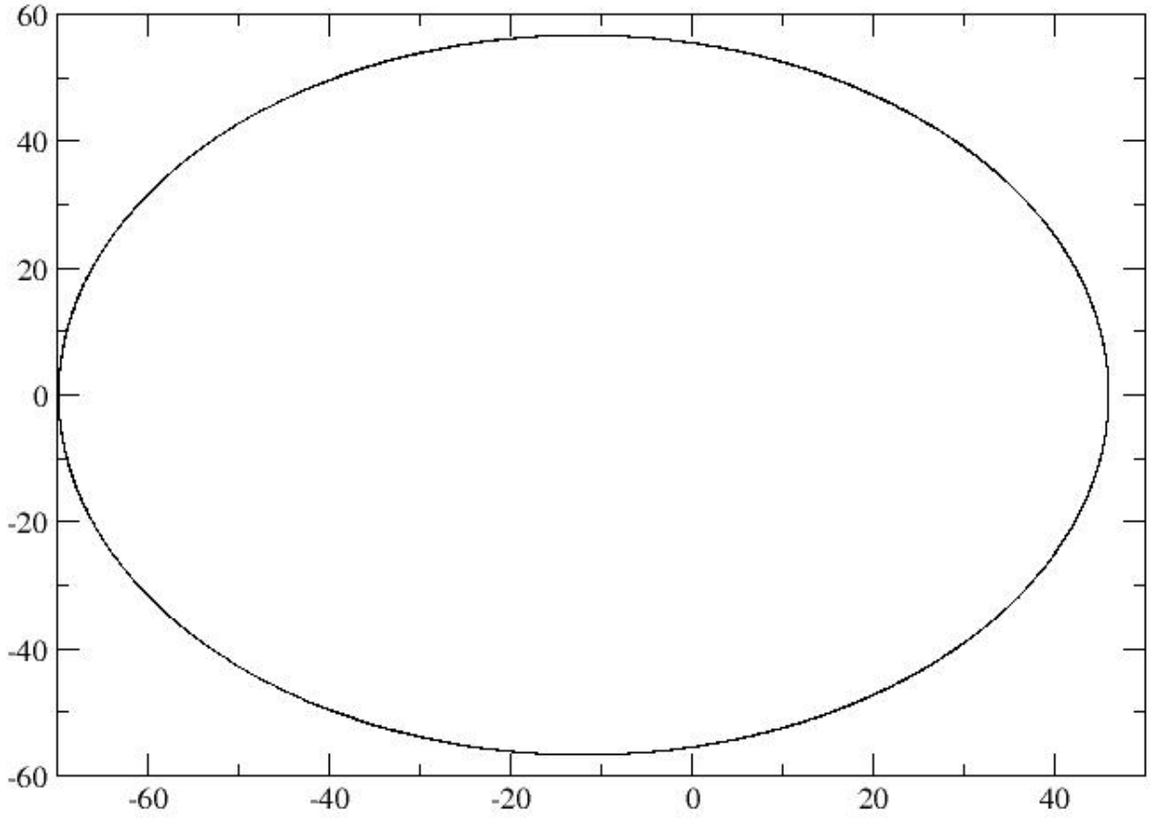
```
      advance=0.00001115996
      if(nr==1)then
         xold=x
         lastx=0
         oldderivative=1
         return
      endif
      dt=0.1d00

    dt=0.001d00
    !   print *,'diff',x-lastx,x,lastx
    if(x<=lastx) then
       print *, '???',x,lastx
       stop
       return
    endif
    nrpts=(x-lastx)/dt
    !print *,nrpts,lastx,x

    !
    !############PRINTING#####################
    !########################################
    !   print *,'x=',x,'lastx=',lastx,'pts=',nrpts
    if(nrpts<=0)return
    do i=1,lrc
       rccopy(i)=rc(i)
    enddo
    do i=1,lic
       iccopy(i)=ic(i)
    enddo
    do i=1,nrpts
       t=lastx+i*dt
       !print *,oldderivative,derivative(x),x
       if(derivative(t-dt) >0.and.derivative(t)<0)then
          lrccopy=lrc
          zerotime=zeroin(t-dt,t+dt,derivative,0Q00)

          !########################################
          !####This will print the minimum and the angle
          !########################################
          !print *,mod(zerotime,twopi)

          oldderivative=derivative(t+dt)
       endif
       !   lastx=x
       rr=1/contex(1,t,rc,lrc,ic,lic)
!       print *,rr
!       stop
       xx=rr*cos(t)
       yy=rr*sin(t)

       !#####################################
       !###We can print the orbit easily by following command
       !#####################################
       print *,xx,yy

    enddo
    lastx=t
    oldderivative=derivative(t)

    !    print *,'lastx=',lastx
    return
!   close(12)
end subroutine solout
!
!
!cccc  the vector field for
!c van der pol

!#####################################################
!###########Differential Equation Adding
!#####################################################
subroutine vf(n,x,y,f,rpar,ipar)
! --- right-hand side of equ
include 'sample.h'
integer*4 n,ipar
real*16 x,rpar
real*16 y(n),f(n)
!      y(1) = u(r), y(2)=u(phi)
!      phi is the indepndent variable in radiant
!      (u,uv) in Shcw case
!(u'=v , v' = -u + GM/L^2 + 3/2 R_sch u^2 )

!####################VARIABLES###############
!###### alpha = rescale the problem, mass of sun and maximum velocity.
!########################################
real*16 G, c, Mearth, Lm, MSun, vmax, Radius, Lsquared, Rsch, GML2, alpha

!#####################################################
alpha=1d0 ! was 1d4
Mearth = 5.972*1.d24 !(*kg*)
MSun = 1.99 * 1.d30 * alpha/Mearth  !(*kg*) (*Normalized to earth's mass*)
```

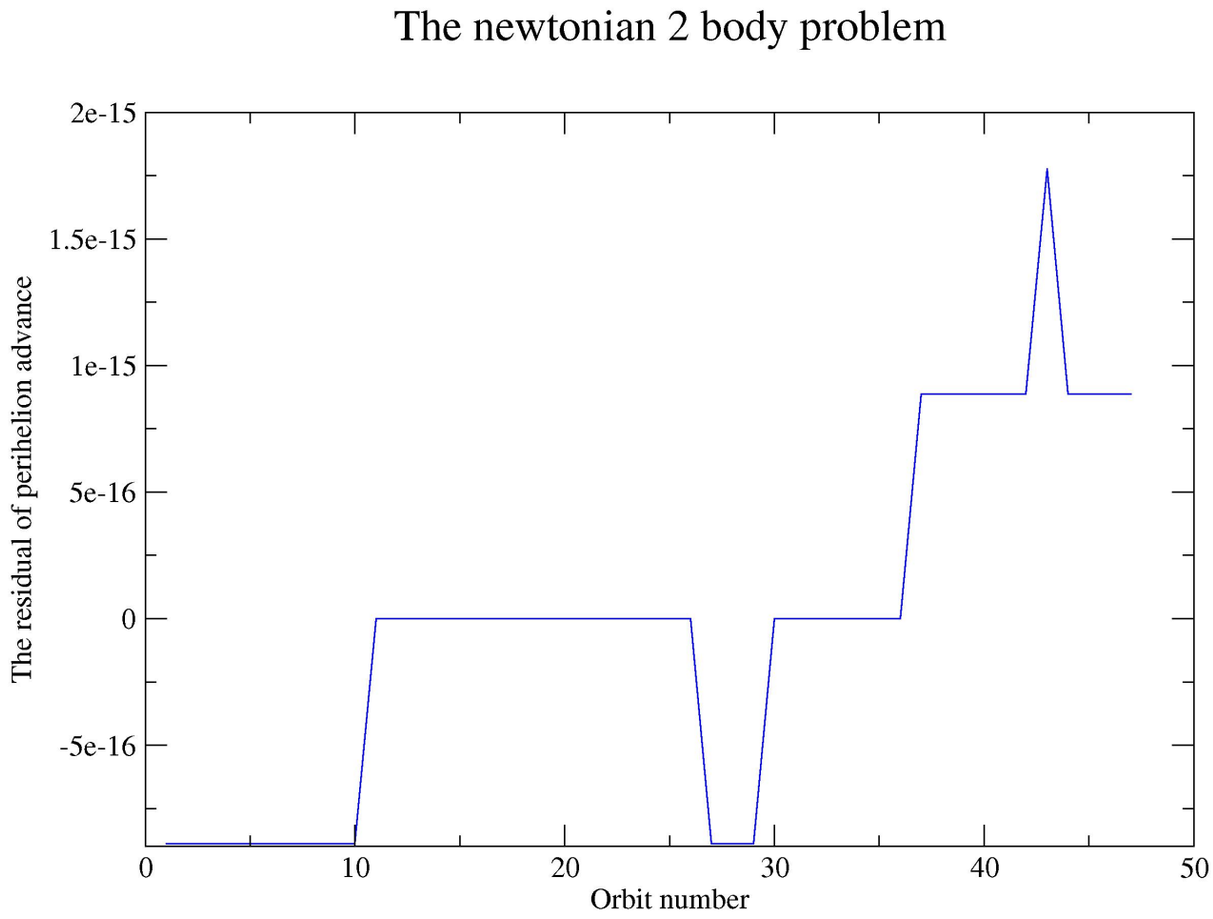
```
Radius=46.0 !(*Giga meter*)
G = 6.67*1.d-26 * Mearth
c = 2.998 * 1.d5
vmax = 58.98* SQRT(alpha)    !(*(Giga meter)/(Mega second)*)
Lm = Radius * vmax;
GML2=G * MSun/(Lm*Lm)
Rsch=2 * G * MSun/(c*c)

!#######################################################

f(1)=y(2)
f(2)= -y(1) + GML2 + 1.5 * Rsch * y(1) * y(1)
!   print *,y(1),y(2),f(1),f(2)
return
end subroutine vf


subroutine dvf(n,x,y,dfy,ldfy,rpar,ipar)
! --- jacobian
include 'sample.h'
integer*4 n,ldfy,ipar
real*16 x,y,dfy,rpar
dimension y(n),dfy(ldfy,n)
return
end subroutine dvf
!#######################################################
!#######################################################


subroutine mass(n,am,lmas,rpar,ipar)
  integer*4 n,lmas,ipar
  real*16 am(lmas,n),rpar

  integer*4 j
!     print *,lmas,n

!########## For stiff equations we can put the am to zero for the relavant variable
  do j=1,n-1
     am(1,j)=1
  enddo
  am(1,n)=1
```

After printing the information of orbit we can plot it with the following command by

```
./make    | xmgrace  -
```

In the xmgrace we can ask for the residual of perihelion in each round and we get the following plot for Newtonian case,


60
40
20
0
-20
-40
-60
-60
-40
-20
0
20
40



As it is clear we do not get perihelion advance for Newtonian gravity with the precision of 10^{-15}, which makes sense.

1.4 x-y plane

Here we do not use all of the symmetries, we just assume that the motion is on a plane and we solve the equation for relative distance between two masses,

$$\frac{d^2\vec{r}}{dt^2} = \frac{-G(m+M)}{|r|^3}\vec{r} \tag{34}$$

We assume $m \ll M$ so we can simplify as,

$$(\dot{v_x}, \dot{v_y}) = \frac{-GM}{(\sqrt{x^2+y^2})^3}\,(x, y) \tag{35}$$

$$(v_x, v_y) = (\dot{x}, \dot{v}) \tag{36}$$

Now we try to get the same result in this new non-simplified coordinate system. In the code we need to assume $n = 4$ which means we have two variables and their derivatives. We also need to have some assumptions for the precision quantities, according to the code below,

```
   ! * * * * * * * * * * * * * * * * * * * * * * * * * * * *
! --- driver for seulex16
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

include 'sample.h'
! --- parameters for radau5 (full jacobian)

! * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
integer*4 n
! Four parameters (x,y,v_x,v_y)
parameter(n=4)
integer*4 nd
integer*4 lwork,liwork,nrdens
parameter (nd=n,nrdens=n)
!  parameter (lwork=4*nd*nd+12*nd+20,liwork=3*nd+20)
parameter (lwork=1000,liwork=1000)
! --- declarations
real*16 y,work
dimension y(nd)
dimension work(lwork)
integer*4 iwork(liwork)
!      vf is vector field
!      dvf is tangent map to vector field (when you compute it analytically
external vf,dvf,solout,mass
real*16 atol,h,rtol
integer*4 i,ijac,imas,iout,itol,mljac,idid,mlmas,mumas,mujac,j
! could use parameters in function
real*16 rpar
integer*4 ipar
integer*4 ifcn

!## x is an independent variable
real*16 x,xbeg,xend
!      character*40  arg
!      call getarg(1,arg)
!      read(arg,'(f7.4)')q2
!      call getarg(2,arg)
!      read(arg,'(f7.4)')p2
!      call getarg(3,arg)
!      read(arg,'(f7.4)')p3
! --- dimension of the system
!      n
! --- compute the jacobian analytically
ijac=0
! --- jacobian is a full matrix
mljac=n
! --- differential equation is in explicit form
imas=1
mlmas=0
mumas=0
! --- output routine is used during integration
iout=1
! --- DIMENSION OF THE SYSTEM
!         N=4
! --- PROBLEM IS AUTONOMOUS
IFCN=1
! --- COMPUTE THE JACOBIAN ANALYTICALLY
IJAC=0
! --- JACOBIAN IS A FULL MATRIX
MLJAC=N
! --- DIFFERENTIAL EQUATION IS IN EXPLICIT FORM
IMAS=0
! --- OUTPUT ROUTINE IS USED DURING INTEGRATION
IOUT=2
! --- INITIAL VALUES
X=0.0Q0

rpar=0
ipar=0

! * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!End points
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
! --- endpoint of integration
! ------ obligatoire au debut -------
xbeg=0d0
xend=100q0
! --- initial values
x=xbeg

do i=1,n
   y(i)=0
enddo

! * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!FILE OPEN
```

```
  ! * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  !!  open (unit = 12, file = "schw_txy.dat")

  ! * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  !Initial values FOR SCHWARZCHILD,
  ! we start from perihelion values
  ! * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  !#### y(1) = x
  !#### y(2) = y
  !#### y(3) = x' =v_x =
  !#### y(4) = y' =v_y =
  !#### we start from y(1)=x= r_mercury and let other variables to be 0
  !#### alpha rescales the problem, masses and ...
  !real*16 alpha , vmax

  ! * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  !alpha=1d0
  y(1)=46.q0 !(Giga meter)
  y(2)=0.q0
  y(3)=0.q0 !(x'=0)
  y(4)= 58.98q0  !(y'= vmax = 58.98 for mercury;//(*(Giga meter)/(Mega !second)*)(*vmax=vmax(1/10^9)(*(Gm/m)*)(10^5/1)*)(*(s/d)*)(*Gm/d*))

  ! --- required tolerance
  !       rtol=1.0q-20
  !       atol=1.0q-7*rtol
  rtol=1.0q-20
  atol=1.0q-29
  itol=0
  ! --- initial step size
  h=1.0d-4
  ! --- set default values
  do  i=1,20
     iwork(i)=0
     work(i)=0.q0
  enddo
  work(1)=1Q-33
  iwork(6)=nrdens
!  print *,iwork(6)
  iwork(2)=3000000
  DO I=1,NRDENS
     IWORK(20+I)=I
  END DO
  ! * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  !  ! * * ALL FOR THE MAIN LOOP
  ! * * * * * * * * * * * * * * * * * * * * * * * * * * * *

  ! --- call of the subroutine radau5
  call seulex(n,vf,ifcn,x,y,xend,h, &
       rtol,atol,itol, &
       dvf,ijac,mljac,mujac,&
       mass,imas,mlmas,mumas, &
       solout,iout,&
       work,lwork,iwork,liwork,rpar,ipar,idid)
  ! --- print final solution
  !       write (6,99) x,y(1),y(2)
99 format(1x,'x =',f5.2,'    y =',2e18.10)
  ! --- print statistics
  write (6,90) rtol
90 format('       rtol=',d8.2)
  write (6,91) (iwork(j),j=14,20)
91 format(' fcn=',i8,' jac=',i8,' step=',i8,&
       ' accpt=',i8,' rejct=',i6,' dec=',i8,&
       ' sol=',i8)
  stop
end program
!

real*16 function derivative(t)
  implicit none
  real*16 t
  real*16 distance
  integer*4 lrccopy,liccopy
  real*16 rccopy(1000)
  integer*4 iccopy(1000)
  common rccopy,lrccopy,iccopy,liccopy
  integer*4 i
  real*16 dt
  !Time step for time derivative
  dt=1.q-18 ! is this a good value
  derivative=(distance(t+dt)-distance(t))/dt

end function derivative


real*16 function distance(t)
  implicit none
  real*16 t
  real*16 contex
  integer*4 lrccopy,liccopy
  real*16 rccopy(1000)
  integer*4 iccopy(1000)
  common rccopy,lrccopy,iccopy,liccopy
```

```
  external contex
  ! JPE no
  ! you want the derivative of sqrt(x^2+y^2)
  ! so that is
  real*16 temp(10)
  integer*4 i
  do i=1,2 ! since you are in R^2
  !contex gives the i component at time t and other variables should be there!
     temp(i)=contex(i,t,rccopy,lrccopy,iccopy,liccopy)
  enddo
  distance=sqrt(temp(1)*temp(1) + temp(2)*temp(2))

end function distance

subroutine solout (nr,xold,x,y,rc,lrc,IC,LIC,n,rpar,ipar,irtrn)
  ! --- prints solution at equidistant output-points
  ! --- by using "contex", the continuous collocation solution
  include 'sample.h'
  integer*4 lic
  integer*4 ic(lic)
  integer*4 nr,lrc,n,ipar,irtrn
  real*16 y(n),rc(8),yint(n),rpar,advance
!!!!!! jpadded
  common rccopy,lrccopy,iccopy,liccopy
  integer*4 lrccopy,iccopy(1000),liccopy
  real*16 oldderivative
  real*16 rccopy(1000)
  external derivative
  real*16 derivative
  save lastx,oldderivative
  real*16 zeroin
  external zeroin
  !! end jpadded
  real*16 dt,t
  integer*4 nrpts,i,j
  real*16 x,xold
  real*16 contex
  real*16 lastx,zerotime

    real*16 rr,angle
    real*16 twopi
    twopi=2.* 3.14159265358979323846264338327950288419716939937510q0
    advance=0.000011159960q0
    if(nr==1)then
       xold=x
       lastx=0
       oldderivative=1
       return
    endif

  dt=0.01d00
  !  print *,'diff',x-lastx,x,lastx
  if(x<=lastx) then
     print *, '???',x,lastx
     stop
     return
  endif
  nrpts=(x-lastx)/dt
  !print *,nrpts,lastx,x

  !############PRINTING#####################
  !########################################
  !  print *,'x=',x,'lastx=',lastx,'pts=',nrpts
  if(nrpts<=0)return
  do i=1,lrc
     rccopy(i)=rc(i)
  enddo
  do i=1,lic
     iccopy(i)=ic(i)
  enddo
  do i=1,nrpts
     t=lastx+i*dt
     !print *,oldderivative,derivative(x),x
     if(derivative(t-dt) <0.and.derivative(t)>0)then
        lrccopy=lrc
        zerotime=zeroin(t-dt,t+dt,derivative,0Q00)

        !#######################################
        !####This will print the minimum and the angle
        !#######################################
        !print *,mod(zerotime,twopi)
        rr=SQRT( contex(1,zerotime,rc,lrc,ic,lic)*contex(1,zerotime,rc,lrc,ic,lic)&
             + contex(2,zerotime,rc,lrc,ic,lic)*contex(2,zerotime,rc,lrc,ic,lic))
     angle=ATAN(contex(2,zerotime,rc,lrc,ic,lic)/contex(1,zerotime,rc,lrc,ic,lic))
     print *,mod(angle,twopi)
     endif
     oldderivative=derivative(t+dt)

     ! print *,rr

     !#######################################
     !###We can print the orbit easily by following command
```

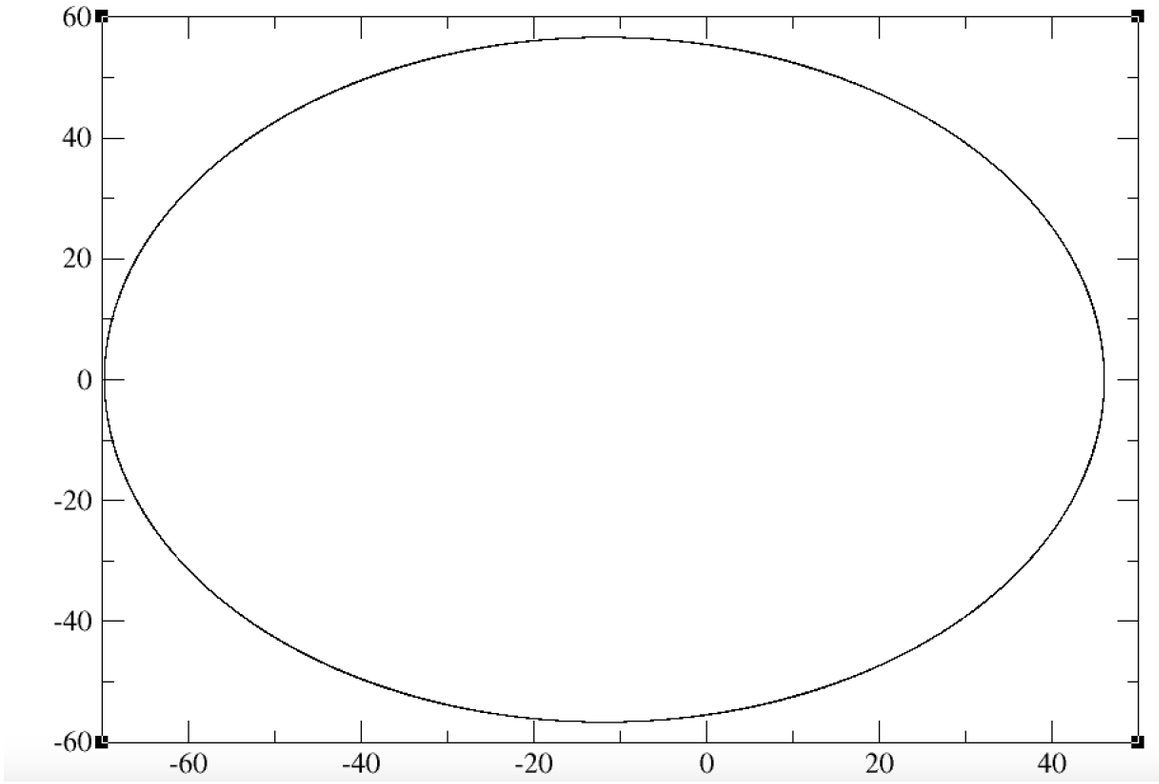
```
      !#######################################
!      print *,contex(1,t,rc,lrc,ic,lic),contex(2,t,rc,lrc,ic,lic)

  enddo
  lastx=t
  oldderivative=derivative(t)

  !   print *,'lastx=',lastx
  return
!  close(12)
end subroutine solout
!
!
!cccc  the vector field for


!################
!###Function |r|^3 = sqrt(x^2+y^2)^3
!################
real*16 function abscubed(a,b)
REAL*16  a, b
  abscubed= (sqrt(a * a + b*b))**3
!  abscubed=a+b
end function abscubed
!#########################################################
!######Differential Equation Adding
!#########################################################
subroutine vf(n,x,y,f,rpar,ipar)
! --- right-hand side of equ
include 'sample.h'
integer*4 n,ipar
external abscubed
real*16 x,rpar,abscubed
real*16 y(n),f(n)
!      y(1) = x, y(2)=y
!      time is the independent
!(x'=v_x , v_x'=  -GMx/SQRT(x^2+y^2)^3, y'=v_y , v_y'=  -GMy/sqrt(x^2+y^2)^3 )

!####################VARIABLES################
!###### alpha = rescale the problem, mass of sun and maximum velocity.
!#########################################
real*16 G, c, Mearth, Lm, MSun, R_perihelion, Lsquared, Rsch, GML2, alpha, vmax


!#########################################################
alpha=1.0
Mearth = 5.972*1.d24 !(*kg*)
MSun = 1.99 * 1.d30 * alpha/Mearth  !(*kg*) (*Normalized to earth's mass*)
R_perihelion=46.0 !(*Giga meter*)
G = 6.67*1.d-26 * Mearth
c = 2.998 * 1.d5
vmax = 58.98* SQRT(alpha)   !(*(Giga meter)/(Mega second)*)
Lm = R_perihelion * vmax;
GML2=G * MSun/(Lm*Lm)
Rsch=2 * G * MSun/(c*c)
!#########################################################

f(1)= y(3) !(x' = v_x)
f(2)= y(4) !(y' = v_y)
f(3)= - G * MSun * y(1)/abscubed(y(1),y(2))    !(v_x' = -GMx/sqrt(x^2+y^2)^3)
f(4)= - G * MSun * y(2)/abscubed(y(1),y(2))    !(v_y' = -GMy/sqrt(x^2+y^2)^3)

!  print *,y(1),y(2),f(1),f(2)
return
end subroutine vf


subroutine dvf(n,x,y,dfy,ldfy,rpar,ipar)
! --- jacobian
include 'sample.h'
integer*4 n,ldfy,ipar
real*16 x,y,dfy,rpar
dimension y(n),dfy(ldfy,n)
return
end subroutine dvf
!#########################################################
!#########################################################


subroutine mass(n,am,lmas,rpar,ipar)
  integer*4 n,lmas,ipar
  real*16 am(lmas,n),rpar

  integer*4 j
!     print *,lmas,n

!########## For stiff equations we can put the am to zero for the relavant variable
  do j=1,n-1
     am(1,j)=1
  enddo
  am(1,n)=1
  return
```

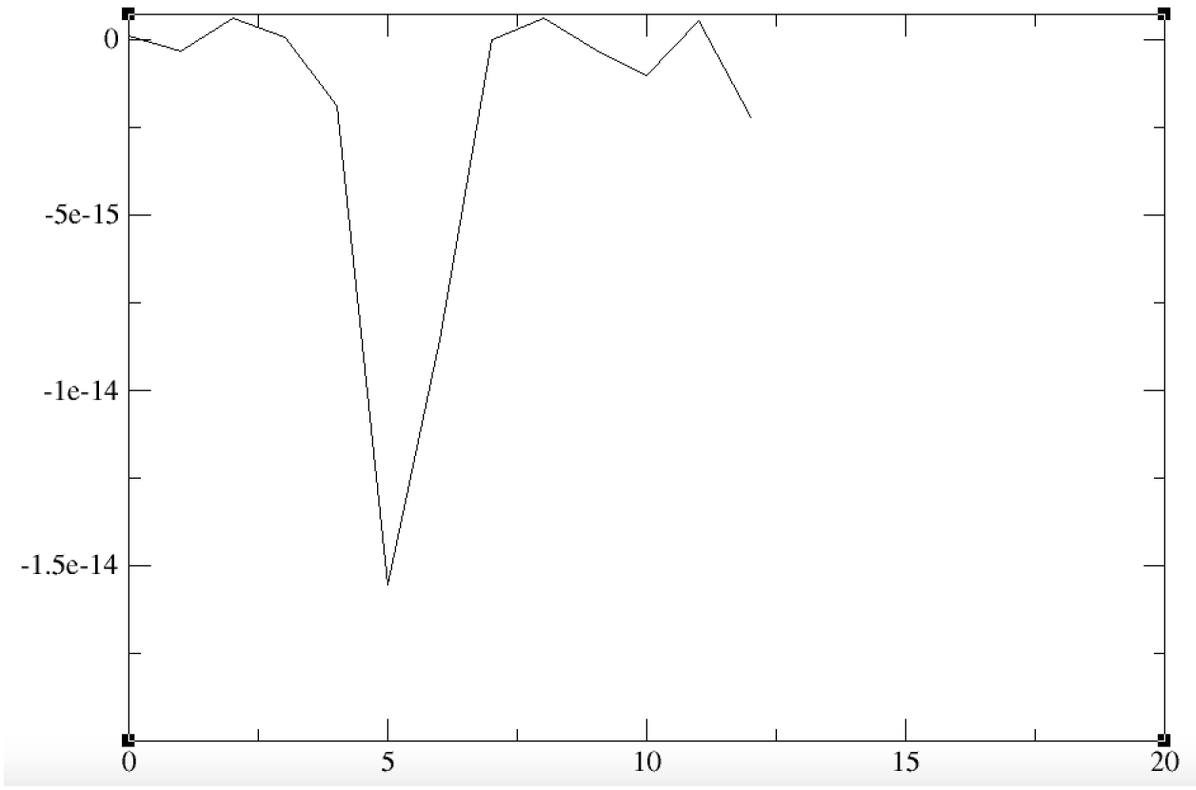
```
end subroutine mass
```

some useful terminal commands, to print the result in a file

```
./makeit_schw_xy > junk.lst
emacs junk.lst
```

For $\alpha = 1$ which is the case for sun and mercury we get the following perihelion and orbit,

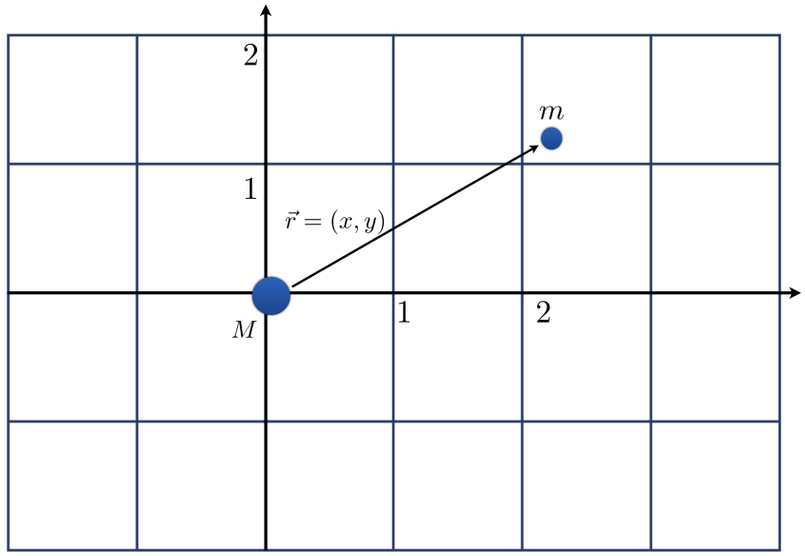




2 Newtonian gravity on the lattice

2.1 2 dimensional case, linear interpolation

For the first step we solve 2D problem, so basically we have,



The potentials of central mass on the cell vertices read,

$$\Phi_{i,j} = -\frac{GM}{\sqrt{i^2 + j^2}} \tag{37}$$

The result of interpolation of potentials at vertices, into the object position is,

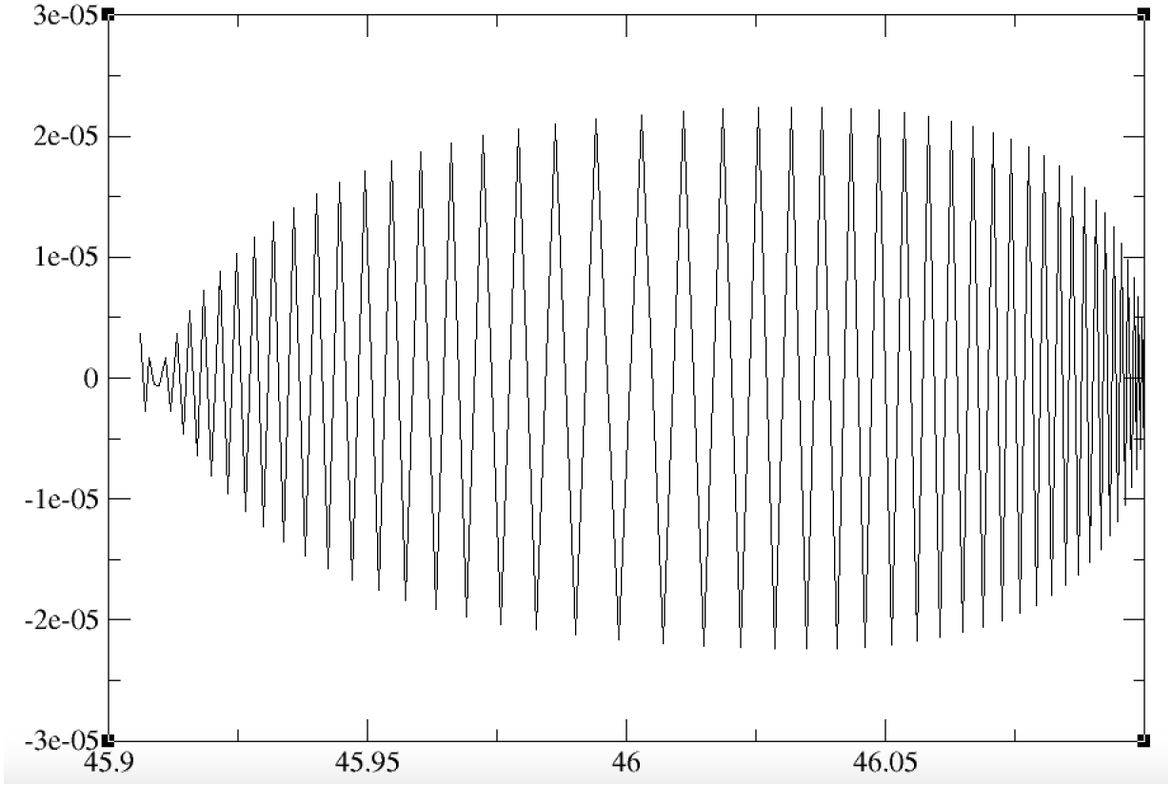
$$\Phi_m(x, y) = \sum_{(i,j) \in 4 \text{ corners}} \frac{\Phi_{i,j}}{2} \left[\sqrt{1 - (x - i)^2} + \sqrt{1 - (y - j)^2} \right] \tag{38}$$

Which we assumed that the interpolation is linear and is ended at the next vertex position, having $\sqrt{1-(x-i)^2}$ which is 0 at $x = i+1$. At the end the acceleration is,

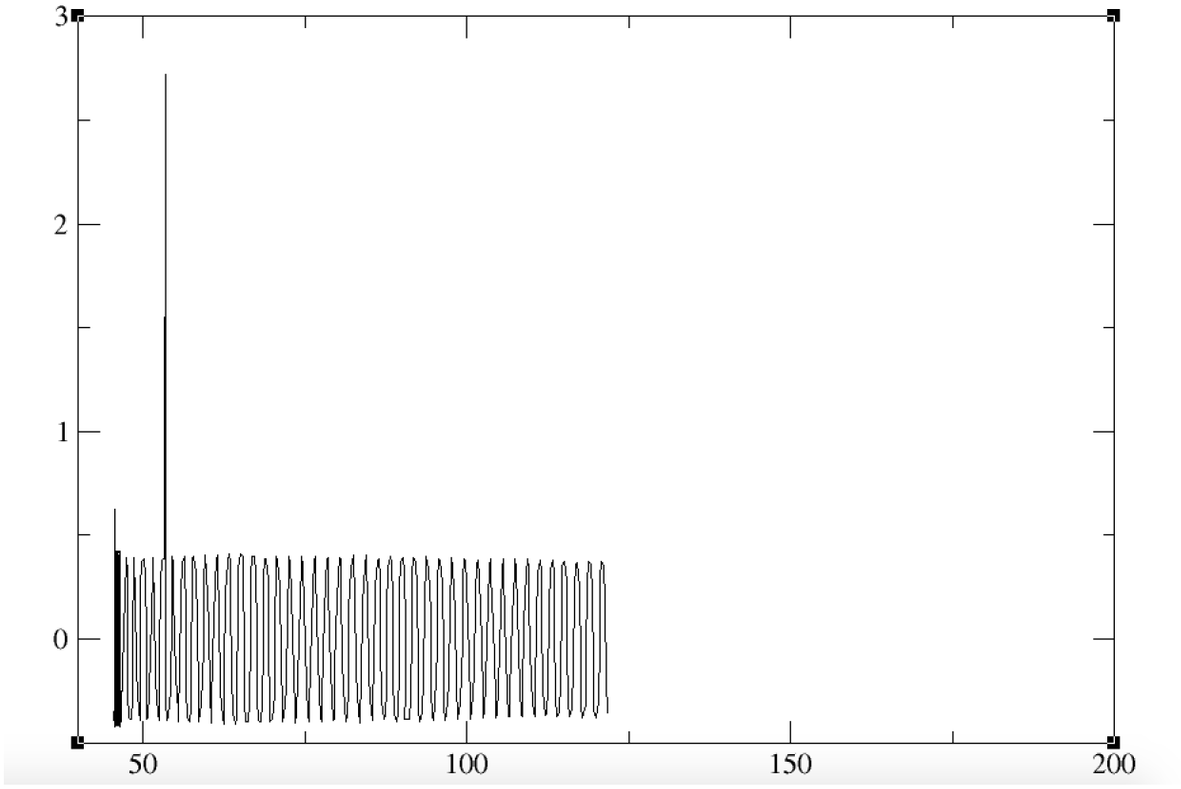
$$\dot{\vec{v}} = \sum_{(i,j)\in 4 \text{ corners}} \frac{\Phi_{i,j}}{2} \left(\frac{x-i}{\sqrt{1-(x-i)^2}}, \frac{y-j}{\sqrt{1-(y-j)^2}} \right) \tag{39}$$

$$\vec{v} = \left(\dot{x}, \dot{y}\right) \tag{40}$$

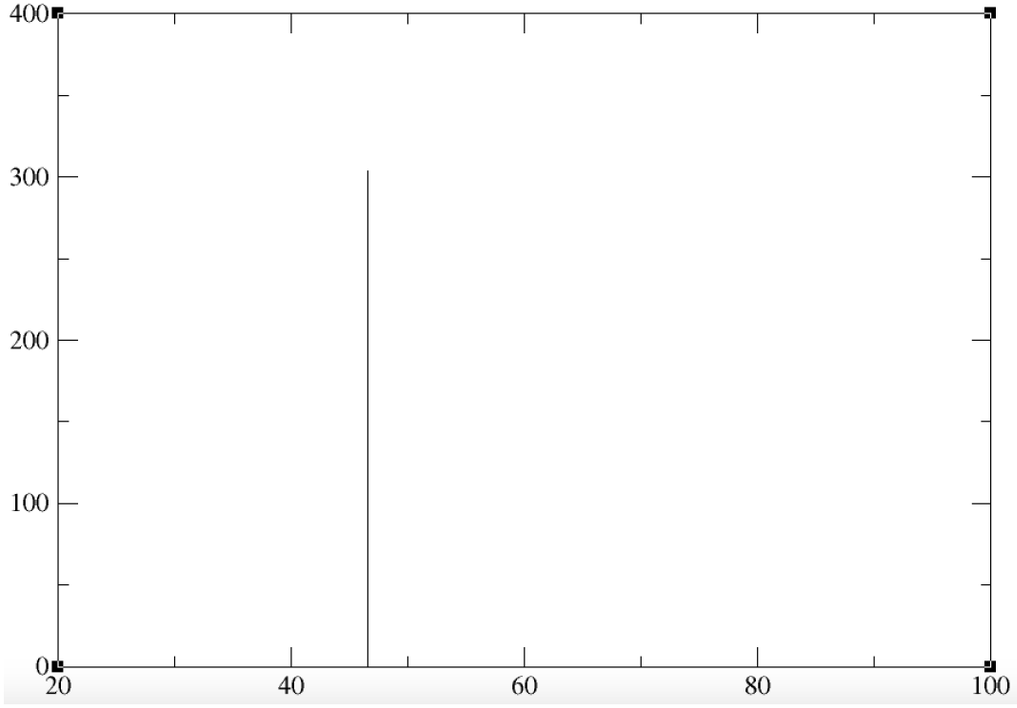
There are some notes at this level, first the situation is singular at the lines $x = i+1, y = j+1$ so we basically we don't expect to get a good result with this interpolation which just cut the forces at the boundaries. But just assume that we just remove singularities by hand in a condition.
What happens is if the particle starts from the initial condition (46,ϵ) which is for mercury because of strong force from the boundaries it just oscillates around the boundary.



For any point in the lattice like $(46.4, 0.4)$.. we see the same oscilation arround the boundary with the amplitude of distance from boundary,

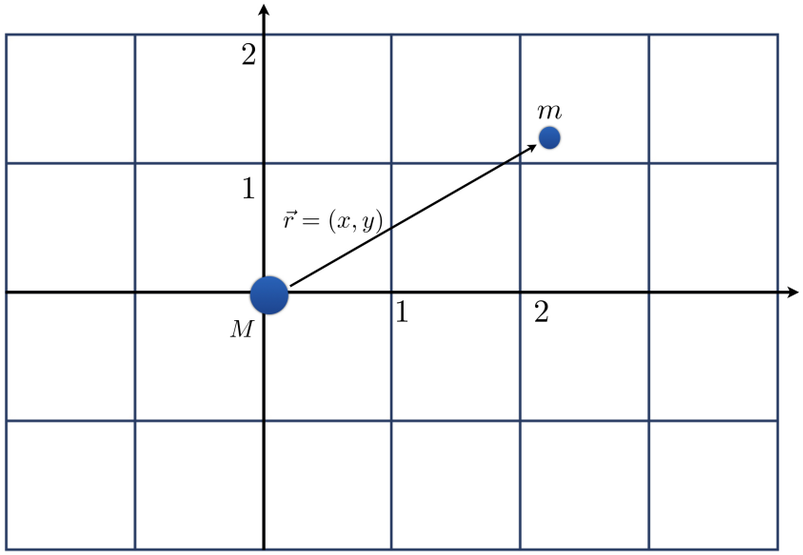


But for the center of the cell meaning $(46., 0.5)$ we see because of symmetry the particle is not attracted to any cell boundaries and just continue to move with the initial velocity.



2.2 2 dimensional case, computing force

For the first step we solve 2D problem, so basically we have,



First we assign masses to the vertices of the cell and the then compute the force from the masses on the particles position,

$$m_{i,j} = \frac{M}{\sqrt{i^2 + j^2}} \tag{41}$$

$$\Phi_m(x,y) = \sum_{(i,j)\,\in\,4\text{ corners}} \frac{-Gm_{i,j}}{\sqrt{(x-i)^2+(y-j)^2}} \tag{42}$$

The mass is assigned to the vertices of the cell according to the distance of the vertex ti the central mass,

$$\dot{\vec{v}} = \sum_{(i,j)\,\in\,4\text{ corners}} Gm_{i,j}\,\left(\frac{x-i}{\Big((x-i)^2+(y-j)^2\Big)^{3/2}},\ \frac{y-j}{\Big((x-i)^2+(y-j)^2\Big)^{3/2}}\right) \tag{43}$$

$$\vec{v} = \big(\dot{x},\dot{y}\big) \tag{44}$$

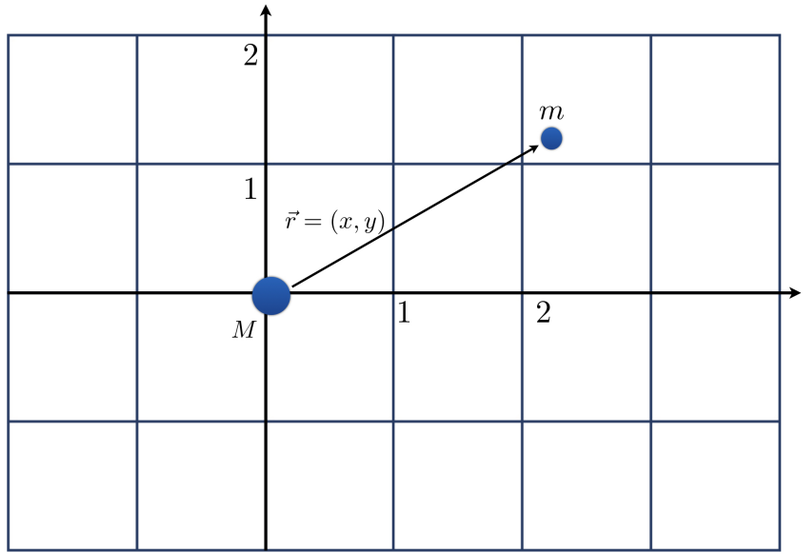
This procedure also has the same problem as before, which in the boundaries we see some bad behaviors,

2.3 2D calculation-Mistakes

Up to now we were computing the force and potential in the position of particle then we took the gradient, but this method does not converge to the right solution. In the correct method we need to compute the gradient on the cell as well. The gradients are defined in the half of each cell and we have, we start from one way derivatives,

$$\nabla\Phi = \frac{\Phi(\vec{x}+\vec{dx}) - \Phi(\vec{x})}{dx} \tag{45}$$

So $\nabla\Phi$ is field in the cell, lives in the half of each side. Again we have the same expressions,



The potentials of central mass on the cell vertices read,

$$\Phi_{i,j} = -\frac{GM}{\sqrt{i^2 + j^2}} \tag{46}$$

The result of gradient of potentials at half of each side reads,

$$(\vec{\nabla}\Phi)|_x = \Big(\Phi_{i+1,j} - \Phi_{i,j}\Big) + \Big(\Phi_{i+1,j+1} - \Phi_{i,j+1}\Big) \tag{47}$$

We have four gradients field defined on the half of the each length, which should be interpolated. The gradients are,

$$\mathcal{F}_{(i+1/2,j)} = \big(\vec{\nabla}\Phi\big)|_{(i+\frac{1}{2},j)},\ \ \mathcal{F}_{(i,j+1/2)} = \big(\vec{\nabla}\Phi\big)|_{(i,j+\frac{1}{2})},\ \ \mathcal{F}_{(i+1,j+1/2)} = \big(\vec{\nabla}\Phi\big)|_{(i+1,j+\frac{1}{2})},\ \ \mathcal{F}_{(i+1/2,j+1)} = \big(\vec{\nabla}\Phi\big)|_{(i+\frac{1}{2},j+1)}, \tag{48}$$

So the interpolated gradient on the central particle position reads,

$$\nabla\Phi_m(x,y) = (\partial_x\Phi_m, \partial_y\Phi_m) \tag{49}$$

$$\partial_x\Phi_m = \Big(\mathcal{F}_{(i+1/2,j)}\big[\sqrt{1-(x-(i+1/2))^2}+\sqrt{1-(y-j)^2}\big]+\mathcal{F}_{(i+1/2,j+1)}\big[\sqrt{1-(x-(i+1/2))^2}+\sqrt{1-(y-j-1)^2}\big]\Big)/(1.91322) \tag{50}$$

derivatives live on the $(m,n) \in \{(i+1/2,j),(i+1/2,j+1),(i,j+1/2),(i+1,j+1/2)\}$. Where 1.91322 is the number to make the wight function normal! Which we assumed that the interpolation is linear and is ended at the next vertex position, having $\sqrt{1-(x-i)^2}$ which is 0 at $x = i+1$.
Better interpolation would be the two dimensional version of CIC method as following,

$$W(x-i,y-j) = \begin{cases} \Big(1-\frac{|x-i|}{L=1}\Big)\Big(1-\frac{|y-j|}{L=1}\Big) & i<x<i+1, j<y<j+1 \\ 0 & \text{others} \end{cases} \tag{51}$$

which is the interpolation rule for a point at (i,j). This is also called area wighted average which according to the area for between point and the source we associate weight and also this interpolation guarantee that the sum over the wights is the area of the cell and for area 1 it is already normal! So for our problem we can write,

$$\partial_x\Phi_m = \Big(\mathcal{F}_{(i+1/2,j)}W(x-i-1/2,y-j)+\mathcal{F}_{(i+1/2,j+1)}W(x-i-1/2,y-j-1)\Big) \tag{52}$$

At the end the acceleration is,

$$\dot{\vec{v}} = -\nabla\Phi_m(x,y) \tag{53}$$

$$\vec{v} = \big(\dot{x},\dot{y}\big) \tag{54}$$

So we can move the particle by the last expression.

2.3.1 Adding grid size

We also can add the cell size to the problem independent of scaling β, because scaling the problem is completely different than solving a problem with different cell size! In fact we add dx to the system which represents the grid size and now the particle is at floor($\frac{r}{dx}$) according to new numbering! So we have the following formulas, The potentials of central mass on the cell vertices read,

$$\Phi_{i,j} = -\frac{GM}{\sqrt{i^2dx^2+j^2dx^2}} \tag{55}$$

where i, j are numbered according to the cell size according to the position of the particle, i.e. $i = floor(r/dx), j = floor(y/dx)$. The result of gradient of potentials at half of each side reads,

$$\big(\vec{\nabla}\Phi\big)|_{x,(i+1/2,j)} = \frac{\Big(\Phi_{i+1,j} - \Phi_{i,j}\Big)}{dx} \tag{56}$$

We have four gradients field defined on the half of the each length, which should be interpolated. The gradients are,

$$\mathcal{F}_{(i+1/2,j)} = \big(\vec{\nabla}\Phi\big)|_{(i+\frac{1}{2},j)},\ \ \mathcal{F}_{(i,j+1/2)} = \big(\vec{\nabla}\Phi\big)|_{(i,j+\frac{1}{2})},\ \ \mathcal{F}_{(i+1,j+1/2)} = \big(\vec{\nabla}\Phi\big)|_{(i+1,j+\frac{1}{2})},\ \ \mathcal{F}_{(i+1/2,j+1)} = \big(\vec{\nabla}\Phi\big)|_{(i+\frac{1}{2},j+1)}, \tag{57}$$

So the interpolated gradient on the central particle position reads,

$$\nabla\Phi_m(x,y) = (\partial_x\Phi_m, \partial_y\Phi_m) \tag{58}$$

$$\begin{aligned}\partial_x\Phi_m = \Big(\mathcal{F}_{(i+dx/2,j)}\big[\sqrt{1-\frac{(x-(idx+dx/2))^2}{dx^2}} + \sqrt{1-\frac{(y-jdx)^2}{dx^2}}\big] + \\ \mathcal{F}_{(i+dx/2,j+1)}\big[\sqrt{1-\frac{(x-(idx+dx/2))^2}{dx^2}} + \sqrt{1-\frac{(y-jdx-dx)^2}{dx^2}}\big]\Big)/(1.91322\times dx)\end{aligned} \tag{59}$$

derivatives live on the $(m,n) \in \{(i+1/2,j),(i+1/2,j+1),(i,j+1/2),(i+1,j+1/2)\}$ and $1.91322\times dx$ is the normalization factor and depends on the dx!
The easier interpolation and more straightforward which is triangle function we have,

$$W(x-i,y-j) = \begin{cases} \Big(1-\frac{|x-i|}{dx}\Big)\Big(1-\frac{|y-j|}{dx}\Big) & i<x<i+1, j<y<j+1 \\ 0 & \text{others}\end{cases} \tag{60}$$

So for our problem we can write,

$$\partial_x\Phi_m = \Big(\mathcal{F}_{(i+1/2,j)}W(x-i-1/2,y-j) + \mathcal{F}_{(i+1/2,j+1)}W(x-i-1/2,y-j-1)\Big) \tag{61}$$

$$\dot{\vec{v}} = -\nabla\Phi_m(x,y) \tag{62}$$

$$\vec{v} = \big(\dot{x},\dot{y}\big) \tag{63}$$

2.3.2 Derivatives on the vertices

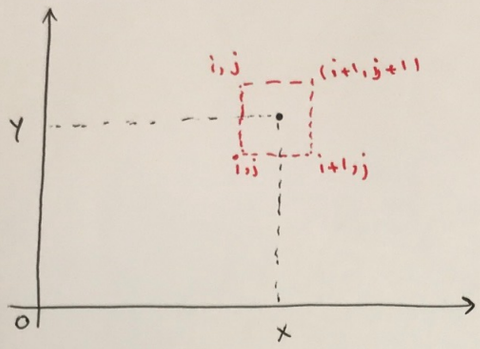
Another way of calculating the force is not having the derivatives on the edges, instead having them on the vertices! But in this case we have to use the value of potential on the other cells and in other words we need 5 cells to compute the derivatives on the vertices completely.

$$\frac{\partial\Phi}{\partial x}|_{(i)} = \frac{\Phi(i+1)-\Phi(i-1)}{2dx} \tag{64}$$

and also we have $\frac{\partial\Phi}{\partial y}|_{(i)}$ at this point! It seems that this method has larger error than the pervious one because we are using more cell to compute a quantity! Or we can take one way derivative!

3 2D correct calculation

Here the interpolation in the line $x = cte$ is just done on the y direction! So we con't care where we define the the vector for gradient, we just care which is line it exist and we interpolate by the direction perpendicular to that line.



$$(i,j) = \left(\text{floor}\left(\frac{x}{dx}\right), \text{floor}\left(\frac{Y}{dY}\right)\right)$$

we compute Φ at each vertices then we interpolate them.

$$\Phi_{i,j} = \frac{-GM_\odot}{\sqrt{i^2dx^2 + j^2dy^2}}$$

The interpolation is linear: $\left.\frac{\partial \Phi}{\partial x}\right|_{bottom} = \frac{\Phi(i+dx,j) - \Phi(i,j)}{dx}$

$$\left.\frac{\partial \Phi}{\partial x}\right|_{top} = \frac{\Phi(i+dx,j+dx) - \Phi(i,j+dx)}{dx}$$

Interpolation:

No matter where we define the vector

$$\Rightarrow \left.\frac{\partial \Phi}{\partial x}\right|_{partic} = \left.\frac{\partial \Phi}{\partial x}\right|_{bottom}\left(1 - \left(Y - \frac{j}{dx}\right)\right) + \left.\frac{\partial \Phi}{\partial x}\right|_{top}\left(Y - \frac{j}{dx}\right)$$

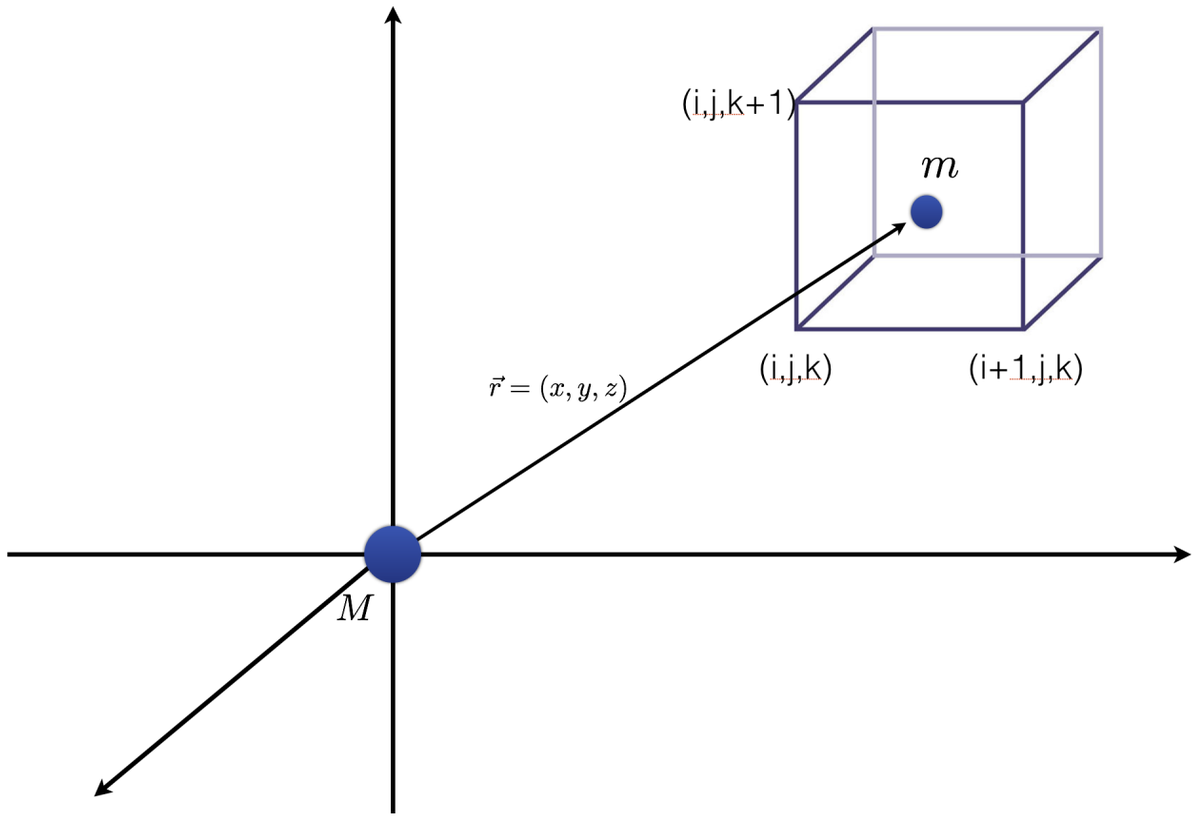
The same for $\frac{\partial \Phi}{\partial y}$

So $\rightarrow$ $\dot{v}_x = -\left.\frac{\partial \Phi}{\partial x}\right|_{partic}$ ✓

$\dot{v}_y = -\left.\frac{\partial \Phi}{\partial y}\right|$

3.1 3 dimensional

The perihelion is computed by taking minimum $r = \sqrt{x^2 + y^2}$ in the code. The problem is illustrated in the figure. As it is clear, the problem is more complicated than before since the effect of each mass on the other is via the lattice vertices, so instead of one source we have basically 4 forces. We make some assumptions to simplify the problem, first we take one of the masses fixed by assuming $M \gg m$, then we set the lattice coordinate by central mass M. Mass m is allowed to freely move on the lattice and the lattice vertices have integer coordinates. Although in certain conditions the angular momentum of the system is conserved and we can define a plane, but we take general configuration first and then we restrict ourselves to some certain configurations.



3.2 Results

Having the central object in the center and the other mass m at position (x, y, z), first we find the lattice which mass m belongs to then the potential of M on the cube vertices around the mass m reads,

$$\Phi_{i,j,k} = -\frac{GM}{i^2 + j^2 + k^2} \tag{65}$$

where (i, j, k) is the position of one cube vertex and is obtained by taking the floor function of mass m coordinate. The potentials are fixed so we only need to interpolate them on the

planet position. The way we do the interpolation is linear, meaning that the potential in the position of m is,

$$\Phi_m(x,y,z) = \sum_{(i,j,k)\in 8 \text{ corners}} \frac{\Phi_{i,j,k}}{2}\left[\sqrt{1-(x-i)^2}+\sqrt{1-(y-j)^2}+\sqrt{1-(z-k)^2}\right] \quad (66)$$

Before writing the expression for the velocities and position updating, we would like to mention that angular momentum vector is not conserved necessarily, probably we can find some situations which the magnitude of angular momentum vector is conserved and it just rotates?! or according to some symmetries we may be able to find a conserved quantity?! does the discretisation respect the symmetries? like conserving energy? probably it highly depends on the numerical method which update quantities in time!
The acceleration on mass m is obtained by taking the gradient of the potential,

$$\dot{\vec{v}} = -\hat{e}_x\vec{\nabla}_x\Phi_m(x,y,z) - \hat{e}_y\vec{\nabla}_y\Phi_m(x,y,z) - \hat{e}_z\vec{\nabla}_z\Phi_m(x,y,z) \quad (67)$$

Since the lattice vertices are fixed we need to just take the gradient of weighting parts,

$$\dot{\vec{v}} = \sum_{(i,j,k)\in 8 \text{ corners}} \frac{\Phi_{i,j,k}}{2}\left(\frac{x-i}{\sqrt{1-(x-i)^2}}, \frac{y-j}{\sqrt{1-(y-j)^2}}, \frac{z-k}{\sqrt{1-(z-k)^2}}\right) \quad (68)$$

Moreover,

$$\vec{v} = \left(\dot{x},\dot{y},\dot{z}\right) \quad (69)$$

Now lets start from some simple configurations, and try to obtain the results. For simplicity we consider a symmetric initial condition which the source M and mass m are both at the $z=0$ plane and based on symmetry of the problem we have no motion in z direction, setting the initial conditions as mercury like object around the sun.
In the code we first find the lattice which the particle belongs to and then we compute the potential on the lattice vertices and interpolate on the mass position.

4 General relativity

The metric in a FLRW universe in Poisson gauge is,

$$ds^2 = g_{\mu\nu}dx^\mu dx^\nu = [-c^2(1+2\Psi)d\tau^2 - 2B_i dx^i d\tau + (1-2\Phi)\delta_{ij}dx^i dx^j + h_{ij}dx^i dx^j], \quad (70)$$

We can parameterize the scalar quantities in metric and write them as $e^{-2\Phi'}$ and $e^{+2\Psi'}$;

$$ds'^2 = g'_{\mu\nu}dx'^\mu dx'^\nu = [-c^2 e^{+2\Psi'}d\tau'^2 - 2B'_i dx'^i d\tau' + e^{-2\Phi'}\delta_{ij}dx'^i dx'^j + h'_{ij}dx'^i dx'^j], \quad (71)$$

The metric and Einstein equations in both parametrization are the same at linear order of perturbation and we have,

$$\Psi'(\vec{x'},\tau') = \Psi(\vec{x},\tau), \quad \Phi'(\vec{x'},\tau') = \Phi(\vec{x},\tau), \quad ds'^2 = ds, \quad x'^\mu = x^\mu \quad (72)$$

For the scalar degrees of freedom which is the case in Schwarzschild metric we can write;

$$ds^2 = [-(1+2\Psi)c^2 d\tau^2 + (1-2\Phi)\delta_{ij}dx^i dx^j], \quad (73)$$

In new parameterization we have;

$$ds'^2 = [-e^{+2\Psi'}c^2 d\tau'^2 + e^{-2\Phi'}\delta_{ij}dx'^i dx'^j], \tag{74}$$

If we expand the coefficients in weak field approximation we obtain equation [73]. Here we try to obtain Einstein equations and Geodesic equation for new parameterization.
From now on we drop the prime " $'$ " from the new parameterization's coordinates but we should notice that the coordinate and the metric are different in the two parameterization.

4.1 Einstein's equations

In the background level the Einstein equations for both metric are similar.

4.2 The geodesic equations, First order

To obtain first order geodesic equations we start from the classical action of massive test particle,

$$\mathcal{A} = -m_o \int \sqrt{-g_{\mu\nu}\frac{dx^\mu}{d\tau}\frac{dx^\nu}{d\tau}}d\tau = \int \mathcal{L}d\tau \tag{75}$$

Where m_o is the mass of the object. If we use the metric definition [73] we can write;

$$\mathcal{L} = -m_o c\sqrt{1-(v/c)^2}\left(1+\frac{2\Psi+2(v/c)^2\Phi}{1-(v/c)^2}\right)^{1/2}$$

The particle Lagrangian at linear order in perturbative potentials is,

$$\mathcal{L} = -m_o c\sqrt{1-(v/c)^2}\left(1+\frac{\Psi+(v/c)^2\Phi}{1-(v/c)^2}\right)+\mathcal{O}(\Phi^2,\Psi^2) \tag{76}$$

Where $v^2 = v_i v^i$, $v_i = \delta_{ij}v^j$ and $v_i = \dfrac{dx_i}{d\tau}$ is the three velocity of particle.
The canonical conjugate momentum q_i defined as $q_i = \dfrac{\partial\mathcal{L}}{\partial v^i}$;

$$q_i = \frac{m_o}{\sqrt{1-(v/c)^2}}\left[v_i\left(1-2\Phi-\frac{\Psi+(v/c)^2\Phi}{1-(v/c)^2}\right)\right]+\mathcal{O}(\Phi^2,\Psi^2) \tag{77}$$

Just note that at the first order perturbation, for small enough v we have $q_i = mv_i$.
We can rephrase the expression and get an equation for v_i,

$$v_i = \frac{dx_i}{d\tau} = \frac{q_i}{\sqrt{q^2+m_o^2}}\left[1+\Psi+2\Phi+\frac{-q^2}{q^2+m_o^2}\Phi\right]+\mathcal{O}(\Phi^2,\Psi^2) \tag{78}$$

So we found an ODE to find the object's path. To find q_i in the path equation we use the Euler-Lagrange equations $\dfrac{dq_i}{d\tau} = \dfrac{\partial\mathcal{L}}{\partial x^i}$.

$$\frac{dq_i}{d\tau} = -\sqrt{q^2+m_o^2}\left(\Psi_{,i}+\frac{q^2}{q^2+m_o^2c^2}\Phi_{,i}\right)+\mathcal{O}(\Phi^2,\Psi^2) \tag{79}$$

Where $\Phi_{,i} = \partial\Phi/\partial x^i$. Using equations [78] and [79] and applying Runge-Kutta algorithm, we can obtain (x, y) position of particle in time. We should notice that the geodesic equations are only valid to first order metric perturbations, but in principle we can define different parameterization of the metric and take the particle motion according to approximated geodesic equation. The exact derivation of equations is in appendix **??**.
In the next subsection we obtain different linearized metric perturbation for Schwarzschild metric.

4.3 Schwarzschild metric

Schwarzschild metric in isotropic coordinate is ,

$$ds^2 = -\left(\frac{1-\frac{r_S}{4r}}{1+\frac{r_S}{4r}}\right)^2 d\tau^2 + (1+\frac{r_S}{4r})^4\ [dr^2 + r^2 d\Omega^2], \tag{80}$$

If we compare the metric with equations [73], [74] we can derive the potentials for Schwarzschild case;

$$\Psi_{exact} = -\frac{1}{2} + \frac{1}{2}\left(\frac{1-\frac{r_S}{4r}}{1+\frac{r_S}{4r}}\right)^2 \tag{81}$$

$$\Phi_{exact} = \frac{1}{2} - \frac{1}{2}(1+\frac{r_S}{4r})^4 \tag{82}$$

where Φ_{exact} and Ψ_{exact} are the exact potentials.
In the weak field approximation we have $\Phi, \Psi \ll 1$ which is equivalent to $r \gg r_S$, so we can expand the expressions in terms of $\frac{r_S}{4r}$;

$$\Psi_{approx} \simeq -2\frac{r_S}{4r} + 4(\frac{r_S}{4r})^2 - 6(\frac{r_S}{4r})^3 + 8(\frac{r_S}{4r})^4 + O\left((\frac{r_S}{4r})^5\right) \tag{83}$$

$$\Phi_{approx} \simeq -2\frac{r_S}{4r} - 3(\frac{r_S}{4r})^2 - 2(\frac{r_S}{4r})^3 - \frac{1}{2}(\frac{r_S}{4r})^4 + O\left((\frac{r_S}{4r})^5\right) \tag{84}$$

For the other parameterization of the perturbed metric, equation [74], we have;

$$e^{+2\Psi'} = \left(\frac{1-\frac{r_S}{4r}}{1+\frac{r_S}{4r}}\right)^2 \tag{85}$$

$$e^{-2\Phi'} = (1+\frac{r_S}{4r})^4 \tag{86}$$

If we expand the expressions for $\Phi', \Psi' \ll 1$ or equivalently $r \gg r_S$;

$$\Psi_{param} \simeq -2\frac{r_S}{4r} - \frac{2}{3}(\frac{r_S}{4r})^3 - \frac{2}{5}(\frac{r_S}{4r})^5 + O\left((\frac{r_S}{4r})^6\right) \tag{87}$$

$$\Phi_{param} \simeq -2\frac{r_S}{4r} + (\frac{r_S}{4r})^2 - \frac{2}{3}(\frac{r_S}{4r})^3 + \frac{1}{2}(\frac{r_S}{4r})^4 - \frac{2}{5}(\frac{r_S}{4r})^5 + O\left((\frac{r_S}{4r})^6\right) \tag{88}$$

Using different potentials in approximated geodesic equation we can check the accuracy of each by comparing with general relativity solution.

4.4 Geodesic equations-Beyond first order approximation

The exact, non linear self contained geodesic equations based on the exact calculation same as before are,

$$\frac{dq_i}{d\tau} = -m_o\sqrt{1-v^2}\left(\frac{v^2\Phi_{,i}+\Psi_{,i}}{1-v^2}\right)\left(1+\frac{2v^2\Phi+2\Psi}{1-v^2}\right)^{-1/2} \tag{89}$$

$$\frac{dx_i}{d\tau} = \frac{q_i\sqrt{1-v^2/c^2}}{m_o}\frac{\sqrt{1+\frac{2\Psi+2(v^2/c^2)\Phi}{1-v^2/c^2}}}{1-2\Phi} \tag{90}$$

First we obtain a non linear algebraic equation for v^2 from [90],

$$q_iq^i = \frac{m_o^2v_iv^i}{1-v^2/c^2}\frac{(1-2\Phi)^2}{(1+\frac{2\Psi+2(v^2/c^2)\Phi}{1-v^2/c^2})}$$

$$v^2 = \left(\frac{1}{c^2}+\frac{m_o^2(1-2\Phi)^2}{q^2(1+\frac{2\Psi+2(v^2/c^2)\Phi}{1-v^2/c^2})}\right)^{-1} \tag{91}$$

We should be very careful to do the expansion consistently, because in v^2 there are v^2 and q^2 terms which can be any order in terms of potentials. So we should expand all the terms to second order and then simplify the equations. We start from v^2 and then substitute itself into v^2 and repeat the procedure to have an expression for v^2 in terms of q^2 and without v^2 terms. The complete calculations can be found in appendix **??**.

$$v^2 = \frac{q^2c^2}{q^2+m_o^2c^2}\Bigg[1+\frac{2\left(2m_o^2c^2\Phi+m_o^2c^2\Psi+q^2\Phi+q^2\Psi\right)}{m_o^2c^2+q^2}+ \\ +\frac{4\left(3m_o^4c^4\Phi^2+2m_o^4c^4\Phi\Psi+3m_o^2c^2q^2\Phi^2+3m_o^2c^2q^2\Phi\Psi+q^4\Phi^2+q^4\Phi\Psi\right)}{\left(m_o^2c^2+q^2\right)^2}\Bigg] \tag{92}$$

By substituting v^2 in the equations [89] and [90] we have second order geodesic equations. Note that it is not necessary to expand the ODEs after v^2 substitution, because anyway the order of equations do not change and is second order. So we have two ways to get second order geodesic equations results.
The first set of equations are;

$$v^2 = \frac{q^2c^2}{q^2+m_o^2c^2}\Bigg[1+\frac{2\left(2m_o^2c^2\Phi+m_o^2c^2\Psi+q^2\Phi+q^2\Psi\right)}{m_o^2c^2+q^2}+ \\ +\frac{4\left(3m_o^4c^4\Phi^2+2m_o^4c^4\Phi\Psi+3m_o^2c^2q^2\Phi^2+3m_o^2c^2q^2\Phi\Psi+q^4\Phi^2+q^4\Phi\Psi\right)}{\left(m_o^2c^2+q^2\right)^2}\Bigg] \tag{93}$$

$$\frac{dq_i}{d\tau} = -m_o\sqrt{1-v^2}\left(\frac{v^2\Phi_{,i}+\Psi_{,i}}{1-v^2}\right)\left(1+\frac{2v^2\Phi+2\Psi}{1-v^2}\right)^{-1/2} \tag{94}$$

$$\frac{dx_i}{d\tau} = \frac{q_i\sqrt{1-v^2/c^2}}{m_o}\,\frac{\sqrt{1+\dfrac{2\Psi+2(v^2/c^2)\Phi}{1-v^2/c^2}}}{1-2\Phi} \tag{95}$$

On the other hand by substituting v^2 into the geodesic equations and expanding to second order in potentials we obtain the below equations for the object's orbit.

$$\frac{dq_i}{d\tau} = -c\sqrt{m_o^2c^2+q^2}\left(\Psi_{,i}+\Phi_{,i}\frac{q^2}{m_o^2c^2+q^2}\right)+ \tag{96}$$
$$+\,c\left[\frac{m_o^4c^4\Psi\Psi_{,i}-4m_o^2c^2q^2\Phi\Phi_{,i}-m_o^2c^2q^2\Phi\Psi_{,i}-m_o^2c^2q^2\Phi_{,i}\Psi+2m_o^2c^2q^2\Psi\Psi_{,i}-3q^4\Phi\Phi_{,i}-q^4\Phi\Psi_{,i}-q^4\Phi_{,i}\Psi+q^4\Psi\Psi_{,i}}{\left(m_o^2+q^2\right)^{3/2}}\right]+\mathcal{O}(\Phi^3..)$$

$$v_i = \frac{q_ic}{\sqrt{q^2+m_o^2c^2}}\left[1+\Psi+2\Phi+\frac{-q^2}{q^2+m_o^2c^2}\Phi\right]+$$
$$\frac{q_ic\left(8m_o^4c^4\Phi^2+4m_o^4c^4\Phi\Psi-m_o^4c^4\Psi^2+8m_o^2c^2q^2\Phi^2+6m_o^2c^2q^2\Phi\Psi-2m_o^2c^2q^2\Psi^2+3q^4\Phi^2+2q^4\Phi\Psi-q^4\Psi^2\right)}{2\left(m_o^2c^2+q^2\right)^{5/2}}+\mathcal{O}(\Phi^3,\Psi^3,..) \tag{97}$$

5 First order geodesic equations and different potentials results

Here we check the accuracy of different potentials in approximated geodesic equation via comparing with the exact GR solution for Schwarzschild metric equations of motion. One interesting test is measuring the perihelion advance in each case and comparing with the GR perihelion advance which is obtained,

$$\delta\theta \approx \frac{6\pi G^2M^2}{c^2l^2} \tag{98}$$

Where $\delta\theta$ is perihelion advance per orbit, l is angular momentum per object's mass and M is central object's mass.
To go in different regime we define a parameter α which scales the physical quantities as following,

$$M \to \alpha M \tag{99}$$
$$v_{per} \to \sqrt{\alpha}v_{per} \tag{100}$$
$$T \to \frac{T}{\sqrt{\alpha}} \tag{101}$$

T is the object's period, v_{per} is the perihelion speed of the object and M is the central object's mass. By α we can go to the strong/weak limit while keeping one period orbit in time $\frac{T}{\sqrt{\alpha}}$.
The initial conditions for the problem are,

$$x_0 = R \tag{102}$$
$$y_0 = 0 \tag{103}$$
$$q_x = R\, v_{per} \tag{104}$$
$$q_y = 0 \tag{105}$$

To compare the results with Mercury observations we use Mercury's facts

$$M_{mercury} = 3.285 \times 10^{24} \text{ kg} \tag{106}$$
$$T = 87.96 \text{ days} \tag{107}$$
$$v_{per} = 58.98\ km/s \tag{108}$$
$$R = 46 \times 10^9\ m \tag{109}$$
$$M_{sun} = 1.99 \times 10^{30}\ kg \tag{110}$$
$$\tag{111}$$

Then we try to plot different figure to compare different potentials. In figure **??** we plot the first and 415th orbit of Mercury like object with coefficient $\alpha = 10^{-2}$ which goes under exact Schwarzschild potentials. it can be seen that we cannot distinguish between two orbits with eye. We choose 415th orbit because in $\alpha = 1$ it is equivalent for 1 century.